Image Format Converter - RAW, JPG, PNG e TIFF

Project Implementation Report

Introdução

Durante a utilização de editores de imagem, constatei que muitos não suportam ficheiros **RAW** ou limitam a quantidade de imagens que podem ser convertidas gratuitamente.

A ideia deste projeto surgiu com o objetivo de criar uma **solução eficiente** de conversão em **massa** de imagens, capaz de lidar com ficheiros **RAW** e outros formatos comuns (**JPG**, **PNG** e **TIFF**), preservando a qualidade e automatizando o processo para pastas inteiras de imagens.

O objetivo final é permitir que fotógrafos, designers e qualquer utilizador possa converter e padronizar grandes volumes de imagens de forma rápida e sem necessidade de recorrer a software pago.

Visão Inicial

O projeto nasceu de uma necessidade pessoal identificada ao utilizar editores de imagens existentes. Constatei que, apesar de existirem editores de elevada qualidade, nenhum oferecia suporte eficiente para arquivos **RAW** em massa e a maioria das soluções online apresentava limitações em termos de quantidade ou exigia pagamento.

A primeira ideia foi criar uma ferramenta simples que permitisse a conversão de imagens **RAW** para **JPEG**, com foco na funcionalidade básica: selecionar uma imagem, processá-la e salvá-la num novo formato. O objetivo inicial era demonstrar que a conversão em lote de **RAW** era possível utilizando ***Python*** e bibliotecas especializadas, como a ***rawpy*** para arquivos **RAW** e a **Pillow** para formatos comuns.

À medida que o projeto avançava, surgiram novas oportunidades de melhoria:

Extensão para outros formatos

Para tornar o projeto mais universal, foi adicionado suporte para os formatos **PNG** e **TIFF**, permitindo a conversão entre **JPEG**, **PNG**, **TIFF** e **RAW**.

Tal exigiu um estudo adicional sobre **modos de cor** e **compatibilidade de paletas** (**RGB**, **RGBA** e **P**), de modo a garantir que as imagens não perdessem canais importantes durante a conversão.

Processamento em massa

Desde o início, era fundamental que a ferramenta fosse capaz de processar **pastas inteiras de imagens**, poupando tempo ao utilizador e aumentando a utilidade do programa.

Tal levou à escolha da biblioteca ***pathlib*** para iterar e manipular ficheiros de forma segura e eficiente.

Interface gráfica simples e funcional

Optei por usar o **Tkinter**, apesar de parecer antiquado, devido à sua simplicidade e compatibilidade com o **Python** padrão.

A ideia era criar uma janela intuitiva com:

- Botão para selecionar uma pasta de entrada;
- Combo box para escolher o formato de saída pretendido;
- Botão para início da conversão;
- Área de logs para feedback em tempo real.

Qualidade e compatibilidade

Desde o início, a prioridade foi manter a qualidade das imagens convertidas.

Para esse efeito, ao converter **RAW**, apliquei correções como o **equilíbrio de brancos** da câmara, saída em **8 bits** por canal e espaço de cor **sRGB**.

Nos formatos comuns, foram **efetuadas conversões de modos de cor**, quando necessário (RGBA → RGB para JPEG, etc.), de modo a evitar a perda de informações ou erros de gravação.

Planeamento modular

O projeto foi estruturado de forma modular, **separando** as funções de conversão, seleção de pasta e registos.

Esta abordagem permite futuras ampliações, como o suporte a novos formatos, filtros de imagem ou a integração com processamento em paralelo para acelerar grandes lotes.

Em suma, a visão inicial evoluiu de uma ideia simples de conversão de **RAW** para um conversor universal e eficiente, capaz de lidar com vários formatos, pastas inteiras e diferentes modos de cor, mantendo a qualidade e a facilidade de utilização para o utilizador final.

Conceitos Técnicos

RAW

Trata-se de um ficheiro bruto gerado diretamente pelo sensor da câmara. Não é comprimido, o que permite preservar **todos** os detalhes de cor e luminosidade captados.

Algumas das vantagens são, a máxima qualidade e flexibilidade para o pós-processamento, bem como a possibilidade de efetuar ajustes finos de exposição, de equilíbrio de brancos e de recuperação de sombras/luzes.

As desvantagens passam pelo facto de os ficheiros serem muito pesados e de cada fabricante de câmaras possuir extensões específicas (**.cr3, .nef, .arw, .dng**), o que gera dependência de bibliotecas especializadas, como a ***rawpy***.

A paleta de cores normalmente armazena informações lineares do sensor que, posteriormente, podem ser convertidas para espaços de cor como o **sRGB** ou o ***AdobeRGB***.

JPEG

Trata-se de um formato comprimido com perdas (***lossy***), muito popular em aplicações digitais.

Apresenta vantagens tal como, **excelente relação** entre a qualidade e o tamanho do ficheiro. Altamente compatível **(funciona em praticamente qualquer dispositivo)**.

Na vertente das desvantagens, a compressão **gera perda de detalhes**, especialmente em áreas com transições de cor suaves. As conversões repetidas podem degradar ainda mais a qualidade.

A paleta de cores é apenas **RGB**, sem suporte para transparência **(não armazena o canal alfa)**.

É ideal para fotografias e imagens em que o tamanho reduzido é mais importante que a fidelidade máxima.

PNG

É um formato sem perdas (***lossless***), com uma compressão eficiente em imagens com áreas de cor uniforme.

Entre as vantagens, destacam-se o suporte de transparência através do canal ***Alpha*** (**RGBA**) e a adequação a imagens gráficas, como logótipos, ilustrações e elementos visuais de UI.

As desvantagens são, arquivos mais pesados que o formato **JPEG** e, para fotografias de alta resolução, não é o mais eficiente em termos de tamanho.

A paleta de cores pode ser **RGB** ou **RGBA**, garantindo uma fidelidade total das cores.

É ideal para gráficos, logótipos e imagens que necessitem de transparência ou preservação total dos pixels originais.

TIFF

Formato versátil e sem perdas, muito utilizado em impressão, design gráfico e fotografia profissional.

As vantagens são muitas mas as mais ponderantes são, a manutenção da qualidade máxima da imagem e o suporte de múltiplas camadas, transparência e diferentes modos de compressão (incluindo sem compressão).

Por outro lado, as desvantagens são os ficheiros **extremamente pesados** e o facto de ser pouco usado em aplicações quotidianas devido ao tamanho.

A paleta de cores é **RGB**, **RGBA**, **CMYK** (para impressão) e até espaços de cores de alta precisão (**16 bits por canal**).

O uso ideal é como arquivo mestre para impressão ou arquivamento profissional de imagens.

NOTAS

Um dos desafios técnicos foi garantir a compatibilidade dos modos de cor com o formato final durante a conversão de imagens.

- O formato **JPEG** não aceita **RGBA**, pelo que tem de ser convertido para **RGB**.
- Os formatos **PNG** e **TIFF** podem trabalhar com **RGBA**, preservando a transparência.
- O formato **RAW**, após o pós-processamento, é convertido para **RGB** ou **RGBA**, a fim de ser guardado nos formatos comuns.

Esta compatibilidade foi tratada no código com condicionais que verificam o modo da imagem (`image.mode`) e aplicam as conversões necessárias (`.convert("RGB")` ou `.convert("RGBA")`).

Bibliotecas Utilizadas

1. Pillow

O **Pillow** é a continuação da antiga biblioteca de imagem do Python (**PIL**). Atualmente, é a **biblioteca mais popular e consolidada para a manipulação de imagens em Python**.

Foi utilizada neste projeto pelo suporte robusto a formatos comuns, como **JPEG**, **PNG** e **TIFF**, pela possibilidade de efetuar conversões de modos de cor, redimensionamento, aplicação de filtros e salvamento em diferentes formatos, bem como pela sua simplicidade e eficiência na manipulação em massa de imagens.

Algumas das funções utilizadas foram:

`Image.open()` - Abre uma imagem e a carrega como um objeto manipulável.

`Image.convert()` - Converte o modo da imagem (por exemplo, **RGBA** → **RGB**, necessário para a compatibilidade com **JPEG**).

`Image.save()` - Guarda a imagem num formato específico, com opções como a qualidade e a compressão.

```
imagem.save(caminho_imagem, format = formato_desejado, quality = 95)
```

Imagem 1 - Exemplos de uso da biblioteca ***Pillow***

```
imagem_raw = Image.fromarray(RGB)
```

Imagem 2 - Exemplos de uso da biblioteca ***Pillow***

2. RawPy

Biblioteca especializada na leitura e processamento de imagens no formato **RAW**. Baseia-se na poderosa biblioteca ***LibRaw***, que interpreta os ficheiros produzidos diretamente pelos sensores das câmaras digitais.

Cada fabricante de câmaras (**Canon**, **Nikon**, **Sony**, etc.) cria um tipo de ficheiro **RAW** específico (**.cr3**, **.nef**, **.arw**, **.dng**). A **RawPy** permite ler todos esses formatos sem depender de software proprietário.

Permite também o processamento posterior controlado: equilíbrio de brancos, aplicação de curvas de gama, profundidade de bits, etc.

Algumas das funções utilizadas foram:

`rawpy.imread()` - Lê uma imagem RAW e cria um objeto manipulável.

`.postprocess()` - Converte um ficheiro **RAW** num array ***NumPy***, num espaço de cores definido (por exemplo, sRGB).

```
with RAW.imread(str(ficheiro)) as raw:
    RGB = raw.postprocess(
```

Imagem 3 - Exemplos de uso da biblioteca ***RawPy***

3. Tkinter

Biblioteca padrão do Python para a construção de interfaces gráficas (**GUI**). Apesar da sua aparência um pouco antiquada, continua a ser amplamente utilizada devido à sua leveza e integração direta com o **Python**, sem necessidade de pacotes externos.

Neste projeto, foi utilizada porque permite criar uma interface intuitiva em que o utilizador pode selecionar uma pasta, escolher o formato de saída e iniciar a conversão sem ter de interagir com o terminal.

Inclui elementos gráficos básicos, como **botões**, **caixas de seleção**, **caixas de texto** e **caixas de diálogo** para seleção de pastas.

Algumas das funções utilizadas foram:

`ttk.Combobox()` - ComboBox usado para selecionar o formato de saída (**PNG**, **JPEG** ou **TIFF**).

`janela.config()` - Permite personalizar a janela.

`janela.mainloop()` - Mantém a janela aberta e escuta os eventos.

```
janela = tk.Tk()
janela.title("Conversor de Imagens - Eliezer Carvalho")
janela.geometry("1200x1200")
#janela.iconbitmap(r"C:\Users\eliez\Desktop\upload_documen
janela.config(bg = "#101820")
```

Imagem 4 - Exemplos de uso da biblioteca ***Tkinter***

4. Pathlib

Trata-se de uma biblioteca **Python** moderna para a manipulação de sistemas de ficheiros e diretórios de forma orientada a objetos. Substitui a utilização da biblioteca tradicional **OS** em muitos cenários.

Algumas das funções utilizadas foram:

`Path()` - Cria um objeto representando diretórios ou arquivos.

`iterdir()` - Itera sobre todos os arquivos dentro do diretório permitindo processar cada imagem.

`saida.mkdir(exist_ok = True)` - Cria uma pasta válida. `exist_ok = True` evita erros caso uma pasta com o mesmo nome já exista.

```
pasta = Path(caminho)
```

Imagem 5 - Exemplos de uso da biblioteca ***Pathlib***

```
for ficheiro in pasta.iterdir():
    if ficheiro.suffix.lower() in extensões:
```

Imagem 6 - Exemplos de uso da biblioteca ***Pathlib***

Explicação Detalhada do Código

`def (converter_imagens)`

1. Criação de uma pasta de saída

```
pasta_saida = pasta_original / f"Fotografias{formato_desejado.lower()}"

pasta_saida.mkdir(exist_ok=True)
```

2. Iteração sobre os arquivos da pasta

```
for ficheiro in pasta_original.iterdir():
        if not ficheiro.is_file():
                continue
    extensão = ficheiro.suffix.lower()
```

Verifica cada ficheiro e ignora subpastas, detetando a extensão para determinar se é **RAW** ou um formato comum.

3. Tratamento de Formatos mais comuns (**JPG**, **PNG** e **TIFF**)

Abre a imagem com `Pillow`, depois converte o modo de cor se necessário:

- **JPG → RGB**
- **PNG/TIFF → RGBA ou RGB**

Guarda com qualidade máxima (95%) no formato desejado e atualiza o log com `log_widget.insert()`.

4. Tratamento do Formato **RAW**

Lê o ficheiro **RAW** com `raw.imread()` e faz um pós-processamento com parâmetros controlados.

```
RGB = raw.postprocess(
   use_camera_wb = True,
      output_bps = 8,
  no_auto_bright = False,
    gamma = (2.2, 4.5),
output_color=rawpy.ColorSpace.sRGB
)
```

Em seguida, converte o array **NumPy** em imagem **PIL** (`Image.fromarray(RGB)`), muda o modo de cor para que seja compatível com o formato de saída, guarda e atualiza o log.

`def (selecionar_pasta)`

Permitir ao utilizador selecionar a pasta de origem através da interface gráfica.

`def (tkinter)`

A janela principal (`Tk()`) configura tamanho, título e cor de fundo.

Os **widgets principais** são: o **botão** "Selecionar Pasta", que chama a função `selecionar_pasta()`; o **campo de entrada**, que exibe o caminho da pasta selecionada; a **etiqueta** e o **combobox**, que permitem selecionar o formato de saída (**JPG,**, **PNG** ou **TIFF**); o **botão** "Converter", que chama a função `converter_imagens()` e passa a pasta, o formato e o log.

Integração com funções:

As variáveis **Tkinter** (`StringVar`) permitem a sincronização entre a interface e as funções do backend.

O ***lambda*** utilizado no botão de conversão garante a correta passagem de parâmetros.

Desafios Encontrados

Ao longo do desenvolvimento do projeto, enfrentei diversos desafios técnicos e conceituais que contribuíram para melhorar a compreensão do problema e a qualidade da solução final.

O primeiro desafio significativo foi compreender e trabalhar com a biblioteca **Tkinter**. Embora seja uma biblioteca padrão do **Python**, a sua abordagem à interface gráfica é relativamente antiga e difere bastante das estruturas de programação modernas. A criação de uma interface intuitiva que permitisse selecionar pastas, escolher o formato de saída e acompanhar o progresso da conversão em tempo real exigiu um estudo detalhado dos **widgets**, das variáveis vinculadas (`StringVar`) e dos métodos de atualização da interface sem bloqueios.

Outro desafio importante foi lidar com a conversão de imagens em diferentes modos de cor. Muitas imagens **RAW** ou em formatos como o **PNG** podem ter canais alfa ou paletas paletizadas que não são compatíveis com o formato **JPG**. Foi necessário implementar verificações de modo (`image.mode`) e conversões adequadas, como **RGBA** → **RGB** para **JPG** e **P** → **RGBA** para **PNG/TIFF**, a fim de garantir que cada imagem fosse convertida corretamente, sem perda de qualidade ou erros de salvamento.

O processamento em massa de ficheiros **RAW** também representou um desafio técnico. Como cada fabricante de câmaras utiliza extensões diferentes (por exemplo, **.cr3**, **.nef**, **.arw** e **.dng**) e diferentes padrões de sensor, foi necessário recorrer à biblioteca ***RawPy*** e à correta configuração do pós-processamento (equilíbrio de brancos, correção de gama e espaço de cores **sRGB**). Garantir que o processo fosse eficiente, preservasse a qualidade e pudesse lidar com grandes pastas sem bloquear a interface foi um dos aspetos mais críticos do projeto.

Por fim, a implementação de um fluxo de conversão em lote robusto, que percorresse pastas inteiras, criasse diretórios de saída automaticamente e registasse logs em tempo real para o utilizador, exigiu atenção ao tratamento de erros, à compatibilidade dos sistemas de ficheiros (**Windows**, **Linux**) e à modularidade do código.

Em suma, todos estes desafios contribuíram para o desenvolvimento de competências em programação, manipulação de imagens e conceção de interfaces, tornando o projeto funcional, fiável e escalável.

Resultados Obtidos

O projeto resultou numa ferramenta eficiente, prática e intuitiva para a conversão em massa de imagens **RAW** e de formatos comuns **JPG**, **PNG** e **TIFF**.

A interface, construída com ***Tkinter***, permite ao utilizador selecionar rapidamente uma pasta, escolher o formato desejado e executar a conversão de forma segura e organizada, sem risco de sobrescrever os ficheiros originais. Cada imagem é processada individualmente, sendo efetuadas verificações de compatibilidade de cores e de modo de imagem, de modo a garantir que a qualidade final seja preservada.

No caso das imagens **RAW**, a ferramenta aplica o pós-processamento adequado, incluindo o equilíbrio de brancos da câmara, a correção de gama e a conversão para **sRGB**, produzindo ficheiros compatíveis e visualmente consistentes. Os ficheiros comuns são convertidos mantendo a fidelidade, sendo ajustado apenas o estritamente necessário para a compatibilidade com o formato de saída.

O registo em tempo real permite acompanhar o progresso de cada ficheiro, informando sobre as conversões concluídas e os possíveis ficheiros ignorados devido a incompatibilidades ou erros, aumentando a confiabilidade e a transparência do processo.

Além disso, o projeto demonstrou modularidade e escalabilidade, podendo ser alargado para suportar novos formatos, aplicar filtros adicionais ou integrar o processamento paralelo, sem ser necessário efetuar grandes alterações à arquitetura existente.

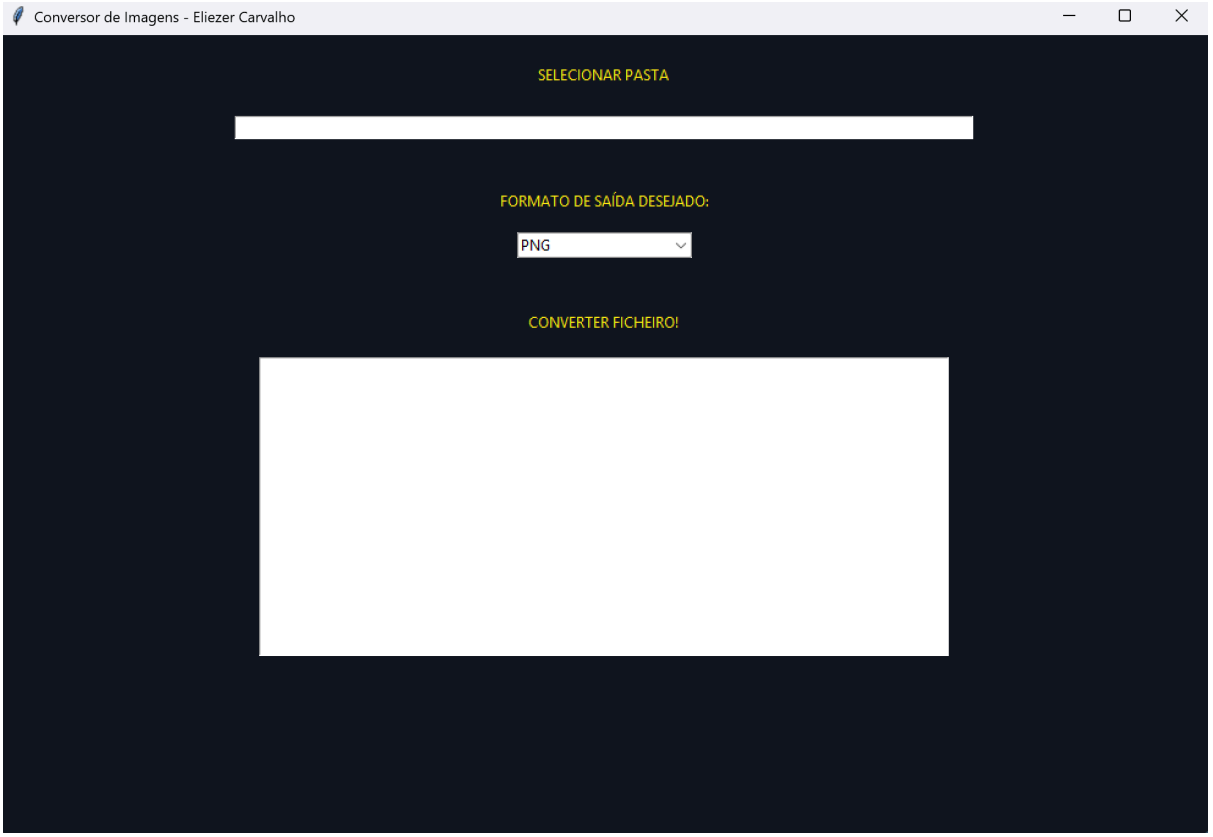

Imagem 7 - Resultado Final

```
from PIL import Image

import rawpy as RAW

import tkinter as tk
from tkinter import ttk, filedialog

from pathlib import Path
```

```
def converter_imagens(pasta_original, formato_desejado, log_widget):

    pasta_original = Path(pasta_original)

    pasta_saida = pasta_original /
f"Fotografias{formato_desejado.lower()}"
    pasta_saida.mkdir(exist_ok = True)


    formatos_comuns = ['.jpg', '.jpeg', '.png', '.tiff']
    formatos_raw = ['.cr3', '.nef', '.arw', '.dng']


    for ficheiro in pasta_original.iterdir():
        if not ficheiro.is_file():
            continue

        extensão = ficheiro.suffix.lower()


        if extensão in formatos_comuns:



                imagem = Image.open(ficheiro)

                if formato_desejado.upper() == "JPEG" and imagem.mode
in ("RGBA", "P"):
                    imagem = imagem.convert("RGB")
```

```
                elif formato_desejado.upper() == "PNG" and imagem.mode
not in ("RGBA", "RGB"):
                    imagem = imagem.convert("RGBA")

                elif formato_desejado.upper() == "TIFF" and imagem.mode
not in ("RGBA", "RGB"):
                    imagem = imagem.convert("RGBA")

                caminho_imagem = pasta_saida /
f"{ficheiro.stem}.{formato_desejado.lower()}"

                imagem.save(caminho_imagem, format = formato_desejado,
quality = 95)

                log_widget.insert(tk.END, f"Conversão Concluída -
{ficheiro.name} → {caminho_imagem.name}\n")

                imagem.close()




        elif extensão in formatos_raw:



                with RAW.imread(str(ficheiro)) as raw:
                    RGB = raw.postprocess(
                        use_camera_wb = True,
                        output_bps = 8,
                        no_auto_bright = False,
                        gamma = (2.2 , 4.5),
                        output_color = RAW.ColorSpace.sRGB
                    )


                    imagem_raw = Image.fromarray(RGB)


                    if formato_desejado.upper() == "JPEG" and
imagem_raw.mode in ("RGBA", "P"):
                        imagem_raw = imagem_raw.convert("RGB")
```

```
                    elif formato_desejado.upper() == "PNG" and
imagem_raw.mode not in ("RGBA", "RGB"):
                        imagem_raw = imagem_raw.convert("RGBA")

                    elif formato_desejado.upper() == "TIFF" and
imagem_raw.mode not in ("RGBA", "RGB"):
                        imagem_raw = imagem_raw.convert("RGBA")

                    caminho_imagem_raw = pasta_saida /
f"{ficheiro.stem}.{formato_desejado.lower()}"

                    imagem_raw.save(caminho_imagem_raw, format =
formato_desejado, quality = 95)

                    log_widget.insert(tk.END, f"Conversão RAW Concluída
- {ficheiro.name} → {caminho_imagem_raw.name}\n")

                    imagem_raw.close()
```

```
def selecionar_pasta():

    pasta = filedialog.askdirectory()
    if pasta:
        entrada_var.set(pasta)
```

```
janela = tk.Tk()
janela.title("Conversor de Imagens - Eliezer Carvalho")
janela.geometry("1200x1200")
#janela.iconbitmap(r"C:\Users\eliez\Desktop\upload_document_content_marketing_seo_digital_guest_post_submission_icon_267785.ico")
janela.config(bg = "#101820")


entrada_var = tk.StringVar()
formato_var = tk.StringVar(value = "PNG")


botão_selecionar_pasta = tk.Button(janela, text = "SELECIONAR PASTA", command = selecionar_pasta, bg = "#101820", fg = "#FEE715",  relief  = "flat").pack(pady = 20)
linha_de_entrada = tk.Entry(janela, textvariable = entrada_var, width = 100).pack(pady = (0, 15))



formato_saída = tk.Label(janela, text = "FORMATO DE SAÍDA DESEJADO:", bg = "#101820", fg = "#FEE715").pack(pady = (25, 15))
opcoes_conversão = ttk.Combobox(janela, textvariable = formato_var, values = ["PNG", "JPEG", "TIFF"], state = "readonly").pack(pady = (0, 15))


botão_conversão = tk.Button(janela, text = "CONVERTER FICHEIRO!", bg = "#101820", fg = "#FEE715", relief = "flat", command = lambda: converter_imagens(entrada_var.get(), formato_var.get(), log_text)).pack(pady = (25, 15))


log_text = tk.Text(janela, height = 15, width = 70)
log_text.pack(pady = (0, 15))

janela.mainloop()
```